4
Evaluation Framework

In the previous chapters, the state of the art for cross layer design techniques has been discussed and a new technique has been proposed based on RF front end impairments. While the benefits of such cross layer techniques have been generally accepted, it is important that such changes are done with caution, as discussed in the paper by [Kawadia and Kumar, 2005]. While making protocol changes that benefit from cross layer design optimisations, it is important to understand legacy issues and maintain backwards compatibility with existing systems. It is also important that any proposed change is specified through a clear programming interface so that the modularity of the architecture is maintained. Various physical layer concerns, such as temporal and asymmetric nature of radio front end impairments, which should be considered while doing cross layer research have been described by [Mandke et al, 2007]. Incorrectly characterizing system losses caused by these impairments can cause the upper layers to adapt and change behavior in a sub-optimal manner, resulting in poor overall performance. These cautionary remarks are not meant to discourage researchers from proposing new cross layer protocol changes; rather they are intended to emphasize the point that the proposed changes have to be very carefully evaluated. To achieve this, it is important that researchers first conduct a thorough theoretical analysis of the proposed protocol change and then implement and validate their idea on an experimentation system. Many such systems have been proposed in the literature, some of which are developed commercially and some which are developed as part of university research have been described in this chapter. However, it is often hard to choose the right experimentation system for a particular use case. To overcome this challenge, an evaluation framework which presents a graphical view of evaluating such systems has been proposed. Researchers can use such a framework to perform trade-off analysis between different requirements and pick a system best optimized for their use-case. Key thesis contributions described in this section are

- A framework to evaluate different experimentation systems in an objective way is presented. Six evaluation metrics, i.e., cost, latency, throughput, hardware agility, software portability, and extensibility, are presented. This framework provides an objective way to evaluate different systems, so that research community can choose the system that is most suited for their use-case. In the absence of this structured approach, it is difficult to pick the system that is best suited for a particular application requirement.
- Survey of several experimental systems is presented with the objective of evaluating these systems on the framework. These systems are compared along the different metrics of cost, latency, and throughput. Results are presented in the form of a table and spider chart, which serve as a visual aid for researchers to understand the strengths and weaknesses of each system along the different metrics and pick the system best suited for their use-case.

4.1 Need for experimentation systems

While the benefits of cross-layer modifications are well known, caution has to be exercised when making these changes and disturbing the highly structured and well architected 7-layer OSI model. In this section, the importance of a good architectural design is discussed. The John Von Neumann architectural for computing systems, the 7-layer Open Systems Interconnect (OSI) model for internet, Shannon's architecture for communication systems, and the plant controller feedback paradigm for control systems are all good examples of successful architectures. All of

these architectures have resulted in longevity of the systems and allowed independent research and development of various modules. Architecture in a system design refers to the breaking down of a system into modular components and systemically specifying the interface between the components. Modularity allows rapid research and development at each layer without researchers at each level having to understand all of the underlying components. Authors in [Kawadia and Kumar, 2005] contend that a good architecture can lead to proliferation of systems. Conversely, an integrated system needs to be hand crafted for every change. Even a small change in one section has ripple effects on the rest of the system and this prevents fast evolution. Such a system eventually reaches innovation saturation. While the von Neuman architecture has proliferated the growth of sequential computing architecture, Valiant points out that the lack of such an architecture for parallel computation is one of the reasons why it has not successfully proliferated [Valiant, 1990]. The 7-layer OSI model is another example of how a good architecture has helped in proliferation of internet and wired architecture. Even the TCP/IP architecture is fundamentally based on the classic OSI model; it only merges some of these blocks into single layer. In his widely cited work on capacity of discrete memoryless channels [Shannon, 2001], Shannon has also made another key contribution. This was the separation of the information source coding from the transmitter error correction coding. This allowed techniques for source coding, such as gzip, rar to evolve at their own pace and on their own preferred system, whereas error correction and modulation schemes evolved at the transmitter side on an implementation system of their choice. All that was needed was a careful specification of a formalized application programming interface (API) between these two layers.

Likewise, feedback architecture is critical in the functioning of control systems design. Control system design is a process that starts with developing a mathematical model that describes the physical system. Next step is to analyze the mathematical model to learn about their dynamic characteristics. The final step is to create a controller to achieve certain dynamic characteristics. A typical control system is shown in Figure 4.1. In this figure, control system refers to the controller, actuators and the sensors. Actuators command the physical plant to take an action and the output of the plant is captured using the sensors. These sensors then provide the feedback signal which is compared with the reference signal to provide appropriate input to the controller. While the actual design of the controller, actuators and sensors will vary with system, this fundamental feedback system architecture forms the core of plant design. It is this fundamental architecture that has enabled the massive proliferation of control systems in wide range of applications. The 7 layer OSI model provided a perfect architecture for wired networks and it has by default become the standard for wireless networks as well. However, wireless networks function differently from wired networks. Using the same 7 layer OSI model for wireless networks is appealing because of its simplicity and abstraction. The multihop decoding and forwarding is simple to understand and implement. However, it has presented its own set of challenges with respect to routing, power control and the need for a medium access control (MAC). There is a fundamental tradeoff between performance and architectural needs. One needs to be careful in pushing for cross layer design without focus on the right architecture underneath as it can lead to brittle behavior. Some general principles to assist in the process of deciding the merits of cross layer design proposals has been presented in [Kawadia and Kumar, 2005].

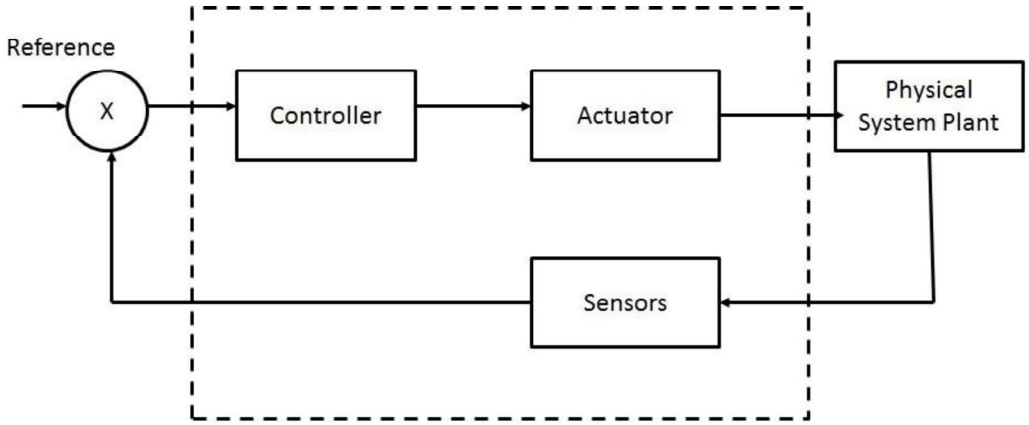

Figure 4.1: Controller System Based on Feedback

First one is to understand the law of unintended consequences. In the field of social sciences, unintended consequences are outcomes that are not the ones originally intended by a purposeful action. Unintended consequences can be positive, negative or perverse. Positive outcome is an unexpected benefit, negative outcome is an unexpected detriment occurring in addition to the desired effect, and a perverse effect is contrary to what was originally intended. There are many examples of unintended consequences, such as medical research [Joan *et al*, 2004], social networks [Bernard *et al*, 2009] and education [Smith *et al*, 1991] to name a few. Likewise, breaking the structured OSI model by allowing cross interactions may lead to unintended consequences, if not handled appropriately [Kawadia and Kumar, 2005]. Dependency graph is the second problem that one needs to keep track of. A dependency graph is a directed graph representing dependencies of various nodes to each other. In a dependency graph, the cycles of dependencies (also called circular dependencies) lead to a situation in which no valid evaluation order exists, because none of the objects in the cycle can be evaluated first. If a dependency graph does not have any circular dependencies, it forms a directed acyclic graph, and an evaluation order may be found by topological sorting. Most topological sorting algorithms are also capable of detecting cycles in their inputs, however, it may be desirable to perform cycle detection separately from topological sorting in order to provide appropriate handling for the detected cycles. Dependency graphs with cyclic dependencies can lead to unstable situations. If a particular parameter is controlled and used by two different adaptation loops, then this could lead to a source of conflict. Adaptive control theory suggests a solution to this problem through the usage of time-scale separation [Johan et al, 2008; Kumar, 1985]. Consider a process C as shown Figure 4.2a, where two processes A and B are trying to control it. If these two control signals are separated in time as shown in the Figure 4.2b, then stability can be achieved. However, this requires detailed and careful analysis. Cross layer design proposals could potentially result in such situations and have to be handled carefully. A final word of caution is related to the development of spaghetti, unstructured code resulting from these cross layer interactions. If not managed properly, it can lead to a system that is not maintainable in the long run. In summary, one can argue that cross layer design proposals are required to advance the art of wireless communications, but a careful approach must be taken to take care of the issues listed above.

Over the past 5 years, active research is being conducted in the area of platforms that enable cross layer design such as the SORA platform [Tan et al, 2011], the beeCube platform [Rotham and Cheng, 2011], MICA platform [Hill and Culler, 2012], testbed for cross layer design in multihop networks [De *et al*, 2005] to name a few. The test platforms which spur active research by its ability to modify or add new capabilities are the ones that truly enhance the capabilities of the research fraternity. Section 4.2 presents a generalized evaluation framework for comparing different experimentation systems. Section 4.3.1 presents the hardware architecture and new

research findings enabled by the Airblue platform, designed by researchers at the Massachusetts Institute of Technology. Section 4.3.2 describes the Wireless Access Research Platform (WARP) designed by researchers at Rice University. Section 4.3.3 describes the HYDRA platform designed by researchers at the University of Texas at Austin. Section 4.4 presents some of the commercially available platforms. Finally, Section 4.5 compares the different systems using the evaluation framework described earlier.

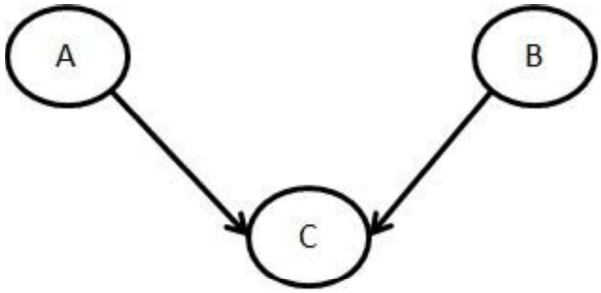

Figure 4.2a: Dependency on Process C from both Process A and Process B

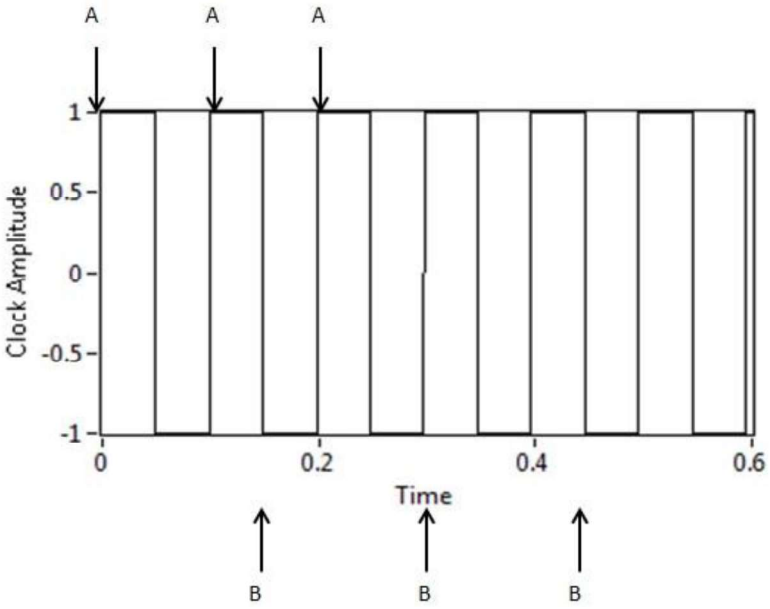

Figure 4.2b: Timescale Separation between Process A and Process B

4.2 Evaluation Framework

The past decade has seen a proliferation of many systems being developed for software defined radio research. Additionally, these systems have also been used for enabling research on cross layer protocols. The research community currently lacks a formalized methodology for objectively comparing such systems. One of the contributions of this thesis is to present an evaluation framework that will address this issue. In this section, six key evaluation metrics that define this framework are described. From a visualization perspective, these metrics are presented in the form of a spider-chart as shown in Figure 4.3. Such visualization graphs make it

easy to compare a particular system with others at a glance. Each metric, as shown on the axis of this graph, has a maximum value of 10 and minimum value of 0. As per this evaluation framework, a higher value for a metric implies a better score for the particular SDR system.

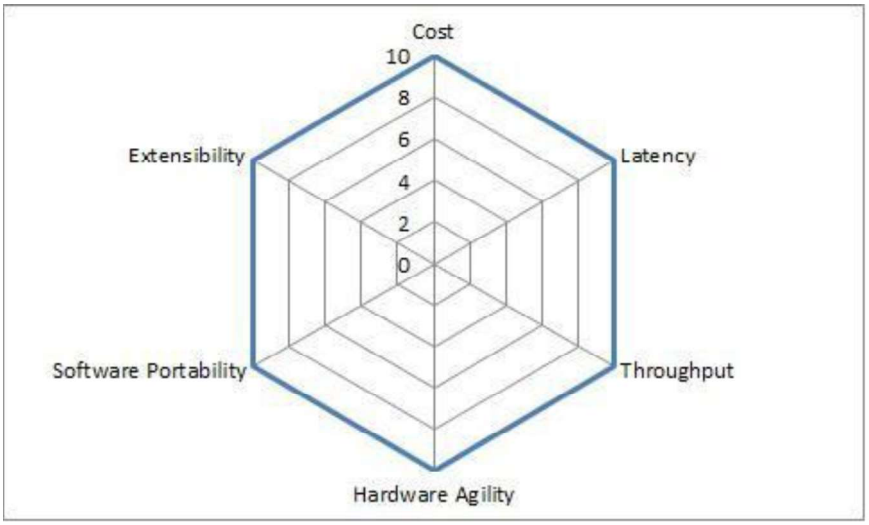

Figure 4.3: Evaluation Metrics

1. Cost: This metric refers to the cost of ownership of the prototyping system. Cost of ownership refers to the cost of purchase, development and maintenance of the system. In the terminology used in this thesis, this does not include the upgrade cost. Ability to upgrade the system and the associated costs is captured independently in a different metric, which is described later. Cost is chosen as one of the key metrics because it defines the affordability of the system. Researchers often have to make a trade-off between functionality and cost. Creators of experimentation systems often add several features to their design. Although these features are useful for many applications, they tend to increase the overall cost of the system, to the extent that it goes out of reach of many researchers. There are various ways in which cost of experimentation system can be kept low, such as use of commercially available off-the-shelf technology and the ability to integrate commodity radios.

2. Latency: Latency is defined as the delay between any two activities in a communication system. One specific example that is commonly used is the turn-around time between transmit and receive nodes. This evaluation metric refers to the smallest value of latency that can be implemented on the prototyping system. An experimentation system should ideally be capable of supporting latencies an order of magnitude better than the minimum latency required by the protocol. Latency is determined by both the speed and deterministic nature of processing. Designers should ensure that their system supports features such as heterogeneous computing [Khokkar *et al*, 1993] and effective bus technology between compute nodes to achieve latency requirements of emerging applications such as cyber-physical systems.

3. Throughput: Throughput, measured in bits/second, is defined as the amount of data transfer that can be sustained on a system. This metric refers to the ability of the system to transmit or receive data at a desired rate. As applications are constantly pushing the need for high streaming data rates, high throughput is fast becoming a very important aspect of communications system design research. It is important that an experimentation system offers users the ability to prototype new algorithms at desired streaming rates. Maximum achievable streaming rates are generally determined by parameters such as instantaneous acquisition bandwidth of the radio

front end, sampling rate of the baseband analog to digital converter (ADC) and the type of heterogeneous processing units used, such as FPGA, DSP, or graphical processing units. It is equally important to think about the type of bus architecture that connects these units together. The type of bus chosen should be able to provide high throughput to meet the demands of latest communication standards.

4. Hardware Agility: This metric refers to the ability of the system to allow users the flexibility to reconfigure input and output terminals during runtime. This capability is applicable for analog characteristics such as radio frequencies, power level, and sampling rate of the ADCs and digital to analog converters (DACs). One application for such frequency agility is in situations where researchers want to conduct experiments in the unlicensed Industrial Scientific Medical (ISM) band due to the availability of low-cost radio front ends in these bands. However, many commercial technologies such as Zigbee, wireless, cordless phones, and Bluetooth also operate in the same frequency range. Any experiment being performed in these bands should be carefully evaluated for possible interference. One approach to avoid this undesired behavior is to identify the region of least interference and tune the RF front end to operate in that region. Hardware agility can also be used for reconfiguring the digital input and output terminals which can be used to transmit a pre-determined word pattern. This can be useful for applications such as bit error rate estimation and pin diode reconfiguration for reconfigurable antenna arrays. An experimentation system should provide users the ability to achieve this through routing of control signals through the FPGA logic.

5. Software Portability: Ability to leverage software developed by research community is a critical element for enabling cross layer research. Software portability metric refers to the ability of the system to allow porting of software across multiple homogeneous or heterogeneous compute nodes. As shown in Figure 4.4, a user of Platform A should be able to easily bring in code developed on Platform B (port-in) and vice-versa (port-out). For this to happen, the experimentation system should offer system design software experience, which allows the software to be specified in various programming languages. For example, in some cases a text based programming language, such as C/C++/Matlab is ideal, whereas in some other cases a procedural language like VHDL could be ideal, and in yet other cases a state machine diagram approach may be more applicable. These various programming paradigms are referred to as hybrid models of computation [Lee *et al*, 1998]. The software support for any ideal experimentation system should be able to handle multiple models of computation so that the developed code is hardware agnostic and portable.

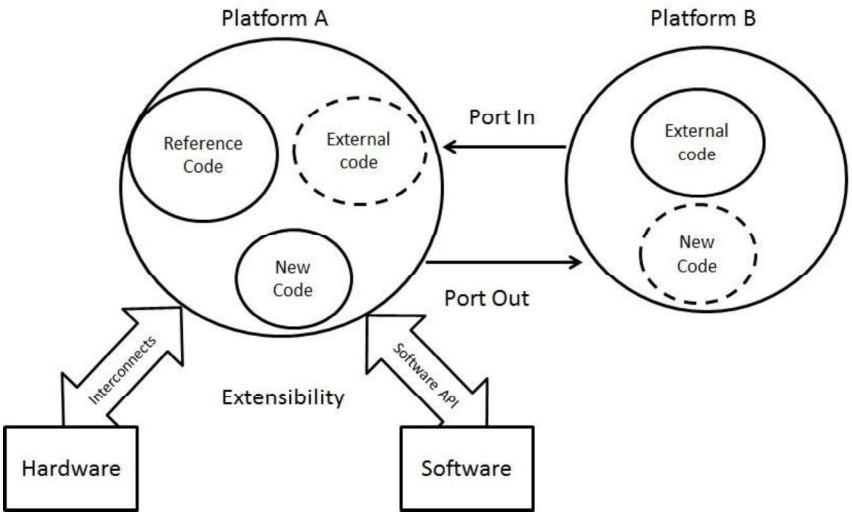

Figure 4.4: Software Portability and Extensibility

6. Extensibility: This metric refers to the ability provided by the system to easily extend (upgrade) its hardware and software capabilities. For good out-of-the-box experience, a system should present a stable start-up codebase. For example, researchers should be able to quickly build an OFDM transceiver by easily putting together basic PHY/MAC protocols either shipped as reference code or available as open source. The next step is to enable algorithm modifications, which may be needed to replace underperforming algorithms with new ones that meet tighter throughput or latency constraints. In some cases, FPGA-specific optimizations may be needed to meet specific resource usage and timing requirements of the FPGA. Such requirements demand extensibility of the hardware and software beyond its original intent. Such type of extensibility is generally enabled by two key features - a generalized hardware interconnect, such as GPIO, SFP/SFP+ or HDMI, which allows users to easily add new hard- ware capabilities to the platform; and a solid software architecture with well-defined and well documented interfaces between different blocks. An Object Oriented Programming (OOP) approach is ideal for such an architecture, as it allows scalability of the programming interface, while maintaining backwards compatibility. The system should also offer stable mechanism for error handling. In summary, software and hardware extensibility is essential to ensure shelf time of the system.

Now that the key evaluation metrics are described, the next section describes various systems found in the literature. These systems have been classified into three categories, which are as follows:

Academic: This category includes systems developed as part of research activities at universities all around the world. While this list is not exhaustive, this thesis focusses on the MIT Airblue system, UT-Austin Hydra system, and RICE WARP system.

Commercial: This category includes systems developed by industry and sold commercially for scientific research activities. This thesis describes the Ettus Research USRP board, Nutaq SDR board, and beeCube miniBee board.

Hobbyist: This category includes low cost boards centralized around a user community of open source enthusiasts. It is primarily targeted towards hobbyists. Examples in this category include systems such as hackRF [HackRF, 2014] and bladeRF [BladeRF, 2014].

Section 4.5 evaluates these systems using the evaluation framework described earlier.

4.3 Academic Systems

4.3.1 MIT Airblue Platform

Airblue is a system developed by researchers at the Massachusetts Institute of Technology to aid cross-layer wireless protocol research. The design of this system is based on the principles of modular refinement, latency insensitive design, and data-driven control. Modular refinement is defined as the ability to make changes in one module of the system without having to understand or make changes to the rest of the modules in the system. Latency-insenstivity and data-driven control are two key properties that are essential for modular refinement in a new protocol [Ng *et al*, 2010], which are described next.

Latency insensitivity is a property that enhances modularity of the system such that the design of a particular module does not depend on the time taken by the previous module. One way in which this can be achieved is by the usage of a FIFO buffer interface (handshake) unit between the two modules. This is a big advantage over latency sensitive systems in which one has to understand the timing constraints of the original implementation by purely examining its

design. This is a very difficult process. The ability to synchronize between data and control is the second property needed for modular refinement. Modifications to the protocol to enable cross-layer research require new ways of sending control signals to the lower layers, unlike standard protocols where specific control and configuration paths are embedded in the design. An implicit synchronization can be achieved by calculating the time taken by data channel a-priori and ensuring that the control channel arrives at the right time. Such a system achieves high performance because additional circuitry is not needed to specifically handle synchronization, but makes it hard to add new controls. To overcome this problem, Airblue designers have proposed the technique of data-driven control. Instead of control and data signals being two independent signals, the idea is to embed the control signal with the data signal, as shown in Figure 4.5. The clear benefit of this approach is that synchronization of data and control is explicit and is forced by the packet, rather than relying on the hardware implementation. Such control signals can be embedded in the packet at different granularity levels as well. The concept of data-driven control is not new to hardware or software systems. The Click Modular Software [Kohler *et al*, 2000] uses the concepts of packet annotations to couple data and control packets together. Similarly, this technique is also used in software-defined radio based systems as described in [Nychis *et al*, 2009].

Figure 4.6 shows a block diagram of the system. The FPGA system is divided to run at three clock speeds, 20 MHz, 25 MHz, and 40 MHz. The device interface block is clocked at a speed of 20 MHz and is used to provide a generic interface between the digital baseband and analog RF front end. The next clock speed of 25 MHz is used to run the baseband processing block, the MAC Unit block, and the debug interface unit. The baseband processor implements the PHY layer, converting bit streams to digital baseband signal during transmission. On the receive side, the baseband processor performs the reverse operation.

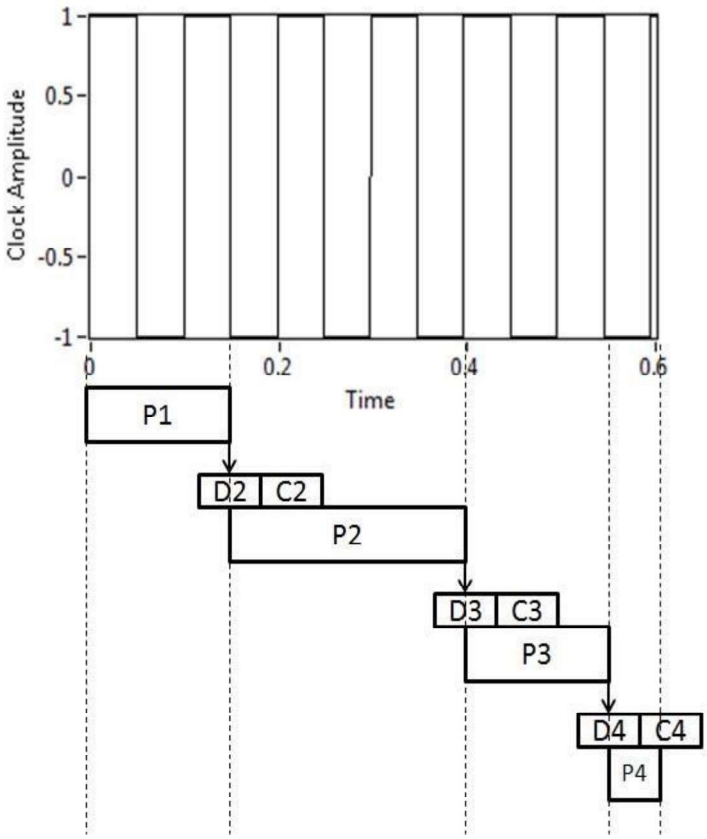

Figure 4.5: Synchronized Data and Control Signals

The MAC unit controls the receiving and transmitting phase of the baseband processor and also implements the sending of ACK signals. The debug interface unit collects other internal signals, representing the state of the system, and communicates this state to the host PC. The soft processor, implemented on the FPGA, runs at 40 MHz and handles off-chip communication signals. The baseband design is built using a set of open source modules from an OFDM Workbench [Ng *et al*, 2007]. The library has been written in Bluespec, which is a high-level design language that compiles into Verilog. It can be further translated into FPGA, ASIC, or software implementations using other tools [Rishiyur, 2004].

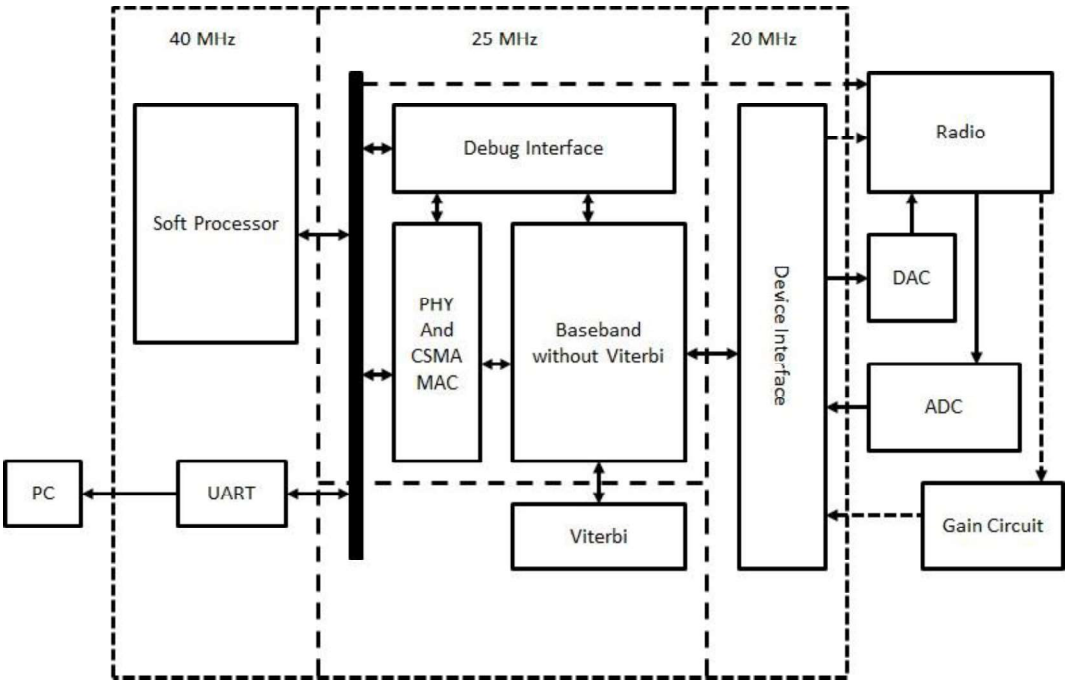

Figure 4.6: Hardware Architecture of the Airblue Platform

The MAC implemented on Airblue has two key features that enable timely communication with the PHY layers. First, it is implemented in hardware, with dedicated low-latency channels to the baseband. Secondly, it is able to communicate with the baseband in a granular fashion, where the granularity is defined at byte level, rather than frame level. This enables the MAC to start processing data as soon as it is available. One of the challenges in a typical RF front end is that components such as the DACs, ADCs, and gain circuits are latency sensitive. If one changes the gain, it takes a few cycles before the correct gain value is reflected in the incoming signals. Additionally, different components demonstrate different timing features. Airblue abstracts the physical layer as a pair of bidirectional FIFOs to which the baseband can connect. The incoming FIFO provides radio samples from the receiver and the outgoing FIFO sends samples to the transmitter. The development platform used for Airblue is Intel's Architect Workbench (AWB), which is an open source management tool [Thomas *et al*, 1988]. AWB provides an interactive environment for configuring, building and running FPGA and general purpose processor co-designs. AWB also makes it easy to debug new modules being developed. In addition to supporting over-the-air operation, Airblue also makes it easy to use an online

channel emulator. Users can connect multiple transceivers to the channel simulator and simulate AWGN and fading environments.

Some of the research activities that Airblue has enabled are described. The modular architecture of Airblue has enabled addition of custom modules, such as the interceptor, to the MAC. Modifications to the baseband PHY to add new blocks to compute and export SoftPHY hints [Jamieson, 2008] have been demonstrated on Airblue. Likewise, replacement of hard output Viterbi decoder algorithm with the BCJR algorithm [Bahl *et al*, 1974] has also been demonstrated on Airblue. All of these examples show the extensibility offered by Airblue to add custom algorithms. Airblue also has the ability to reconfigure the MAC during runtime through interrupts. Such a mechanism is useful in CMAP, where the MAC must first receive the headers of the ongoing transmission, and then switch to transmit mode if its pending transmission does not conflict with the ongoing transmission. Spinal codes [Perry *et al*, 2012] have been implemented on Airblue and it has been demonstrated that these codes can decode up to a rate of 10 Mbps. Spinal codes are a new class of rateless codes that enable wireless networks cope with time-varying channel conditions, without requiring any explicit bit rate selection. Traditionally, hardware designs partitioned across multiple FPGAs have suffered from low performance because of the inefficiency of maintaining cycle-by-cycle timing among discrete FPGAs. [Fleming *et al*, 2012] have presented a mechanism by which complex designs may be efficiently and automatically partitioned among multiple FPGAs using explicitly programmed latency-insensitive links.

4.3.2 RICE WARP System

Wireless open-Access Research (WARP) [Hunter et al, 2006] is a scalable and extensible system, with the ability to be programmed being one of its key features. WARP offers interaction between the PHY and MAC layers through a flexible interrupt driven interface, which enables evaluation of a large class of cross-layer protocols. Figure 4.7 depicts the WARP hardware. Block A depicts the main board which contains the Xilinx Virtex-II Pro FPGA board. This FPGA contains 8 Rocket I/O transceiver blocks, two PowerPC 405 processor blocks, 9,280 logic programmable slices, 88 multiplier blocks and 1,584 KB block memory size. The PHY layer of WARP is based on custom OFDM transceivers, which are intended for IEEE 802.11 a/g/n standards. The MAC protocols are implemented in C and interact with the PHY processing units and supporting peripherals in the FPGA fabric using the flexible interrupt driven interface. Block B represents four general purpose daughter-card slots in the FPGA board, which can be utilized to design radio, analog input/output and video functionality. The custom designed WARP radio is capable of supporting both the 2.4 GHz ISM and 5.8 GHz ISM bands with a 40 MHz bandwidth. Because there are four general purpose daughter-card slots, WARP can be extended to enable 4 x 4 MIMO systems by tuning the radio front end. Block D represents the 10/100 Ethernet port bus interface from the WARP to the PC.

Many practical experiments have been implemented on WARP, including the 2x2 Alamouti and 2x2 spatial multiplexing systems, with antenna selection capability. MAC protocols can be rapidly implemented through a state machine developed in C, which can then be compiled to run on one of the two PowerPC 405 cores. When downloaded onto the PowerPC, the MAC protocol directly communicates with the hardware peripherals. [Hunter *et al*, 2006] have created a flexible interface that provides user-level access to parameters common to most physical layers, in order to facilitate the development of new MAC protocols on WARP. Hence, WARP provides seamless flexibility in the development of cross-layer protocols such as SNR-based rate adaptation, advanced MIMO MAC protocols with beamforming and smart antenna selection features. Different medium access protocols such as ALOHA, Carrier Sense Multiple Access (CSMA), Carrier Sense Multiple Access with Collision Avoidance (CSMA/CA), and Orthogonal Frequency Division Multiple Access (OFDMA) have also been implemented on WARP.

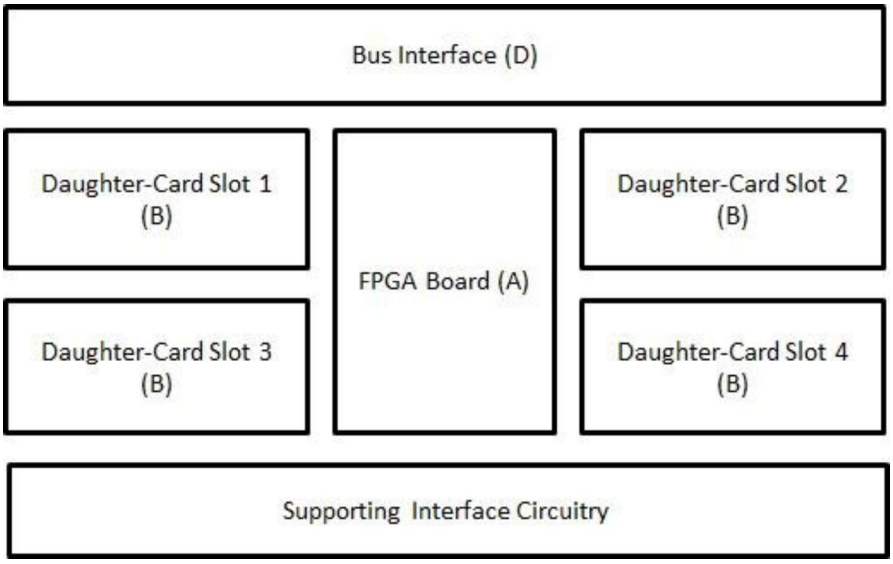

Figure 4.7: Hardware Architecture of the Rice WARP platform

Flexibility to create custom designs has resulted in many significant research findings based on WARP. ArgosV2 is a 64-antenna base station prototype and serves as a compact, powerful, and scalable multi-antenna research platform based on WARP [Shephard *et al*, 2013]. The modular architecture and real-time capability of ArgosV2 can support hundreds of base station antennas and tens of users with streaming applications. Researchers have leveraged the flexibility of WARP to overcome the challenge of extending full-duplex communication for long-range applications due to residual behavior from self-interference [Sahai *et al*, 2013]. [Kivayash *et al*, 2013] have presented the design and implementation of spyware communication circuits built into the widely used carrier sense multiple access with collision avoidance (CSMA/CA) protocol. The spyware has been implemented and evaluated on WARP using metrics such as implementation efficiency of encoder, robustness of communication scheme to heterogeneous CSMA/CA effects, and difficulty of covert channel detection. [Magistretti *et al*, 2012] have used the WARP to design, implement and evaluate a modified version of the IEEE 802.11 protocol (802.11ec) without control messages. Instead of explicit control messages, 802.11ec employs symbol sequences that can be correlated with the timing information. They have used WARP because it gives them the flexibility to perform a large number of experiments and compare the performance of their proposed standard with the existing standard. [Duarte *et al*, 2012] have provided an experiment-driven characterization of a full-duplex wireless system on WARP. Through experimental results, they show that as the received self-interference power increases, so does the average amount of cancellation for active cancellation techniques. They have also derived an experiment-driven model and performed capacity analysis of a full-duplex wireless system implemented using WARP with commercial off-the-shelf MIMO Radios. [Shi *et al*, 2009] have created models for synchronous CSMA (S-CSMA) using WARP. The flexibility of this system allowed them to measure the impact of clock drift on MAC parameters such as contention window size, control packet size and carrier sense regulated by usage of gated time. In addition to the above mentioned research outcomes, a large number of research platforms have also been developed based off WARP [Hershberger *et al*, 2013; Shishkin *et al*, 2011; Korakis *et al*, 2009].

4.3.3 UT Hydra System

Hydra is a fully flexible wireless prototyping system developed at the University of Texas

at Austin [Mandke *et al*, 2007]. Hydra consists of a hardware board and PC-based software stack as shown in Figure 4.8. The software stack is implemented on a general purpose processor and contains implementation for the PHY, MAC and Network layers. FPGA or ASIC implementations

used by other prototypes are high performance, but the trade-off is that they require experience with hardware description languages. To circumvent this problem, Hydra software stack is implemented using open source GNU Radio [Ettus Research, 2014] and Click Modular Software. The modular and open-source nature of these programs makes it easy for researchers to develop individual blocks and then leverage intellectual property (IP) available on the community. Additionally, GNU Radio stack also provides a convenient application programming interface for setting various parameters on the RF front end of Hydra, which is based on the Universal Software Radio Peripheral (USRP) board. The USRP has continuous frequency coverage in the ISM 2.4 GHz and ISM 5.8 GHz bands. The bus interface between the hardware board and software stack is USB 2.0.

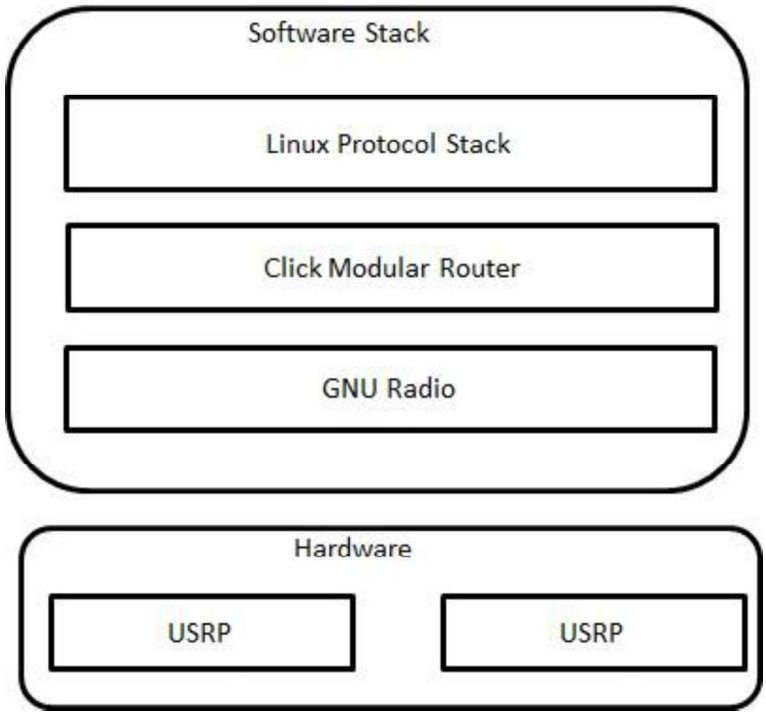

Figure 4.8: Architecture of the UT Hydra platform

The hardware components of USRP include an FPGA, four high-speed ADCs and DACs. The RF front end can be programmed to provide various signal processing functionalities such as filtering, upconversion, and downconversion. Hydra can be further extended for synchronized multiple input multiple output (MIMO) capability by adding more USRP RF front ends.

Some of the key research findings that have been enabled by Hydra are described next. [Mandke *et al*, 2007] have suggested temporal scaling, reciprocity, and cross-layer adaptation as three higher layer considerations that should be kept in mind when designing any cross-layer protocol. Temporal scaling refers to the time scale or granularity with which communication happens between the physical layer and the upper layers. Generally, temporal scaling is measured as a function of the time required to transmit and receive packages. However, it is also important to consider the amount of time that is taken to process the packets. This implies that the MAC layer has to appropriately tune its parameters such as inter-frame spacing, as the packet

sizes change. Secondly, many researchers generally assume that the measurement parameters estimated in the forward and reverse link in a bidirectional communication link are reciprocal.

This implies that if one makes a particular measurement in the forward link, then this measurement can be considered as valid in the reverse link as well. Through various experimental results, [Mandke *et al*, 2007] have shown that the reciprocity assumption may not always hold true. Interference and variance in the RF hardware are the two main reasons that contribute to this asymmetry. In most cross-layer design algorithms, higher layers modify their behavior based on information received from the lower layers. The behavior is however not as simplistic and higher layers must be made aware of the degrees of freedom available, such as automatic gain control mechanisms and impact of RF front end impairments, available at the lower layers.

[Daniels *et al*, 2010] have shown how adaptation can be performed in a convolutionally coded MIMO OFDM wireless system through supervised learning and SNR order. The approach described in this paper has been implemented on Hydra. [Kim *et al,* 2009] have demonstrated an experimental evaluation of rate adaptation for multi-antenna systems. This paper proposes extensions of two well-known link adaptation algorithms, Receiver-Based AutoRate (RBAR) and Auto Rate Fall back (ARF). Through the implementation on Hydra, the practical challenges in MIMO systems resulting from an additional spatial dimension have been demonstrated. [Daniels *et al*, 2009] have shown an online learning framework for link adaptation in wireless networks. Compared with supervised learning, this online framework uses real-time measurements to update the rate-adaptation classifier. [Daniels *et al*, 2008] have presented throughput and delay measurements of limited feedback beamforming in Indoor Wireless Networks. They use Hydra to measure high throughput for various delays and experimentally verifies the exponential relationship between throughput loss and delay.

4.4 Commercial Systems

This section provides a brief overview of some of the commercial systems that can be used for cross layer design. One such example is the commercial SDR platform from Nutaq, Inc. [Nutaq, 2014]. Nutaq SDR boards offer an FPGA and CPU combination as processing elements. For example, the PicoSDR features a Xilinx Virtex-6 FPGA and an embedded Quad-Core i7 processor, whereas the ZeptoSDR features a Xilinx Zynq-7 FPGA and an embedded ARM Cortex-9 processor. This system features a radio frequency front-end, tunable from 300 MHz to 3,800 MHz. Tuning bandwidth is 1.5 MHz to 28 MHz. Data transfer between the onboard and external processors is either through PCI Express or Gigabit Ethernet bus technology on the picoSDR. Both of these models feature a software stack allowing a model-based design approach using Mathworks Simulink and a text-based approach using the GNU Radio open source software. While the picoSDR is capable of only up to 4 x 4 MIMO support, other variants such as µSDR420 Massive MIMO is capable of providing support for 100x100 MIMO configuration. All of the other radio configurations, data transfer, and processing capabilities of the µSDR420 are similar to the picoSDR. The Nutaq website features reference designs for 64-QAM MIMO OFDM and FPGA-based physical layer implementation of 802.15.4 standards. Beecube Inc. [Beecube, 2014] offers cross layer prototyping systems through its reconfigurable platform consisting of scalable, full speed interconnected modules. It supports flexible expansion options through interconnects such as SFP/SFP+ and HDMI. It has a symmetrical 4-FPGA based module architecture that allows for high availability and easy upgrade path for increased capability. It supports both direct and independent interfaces per FPGA. It supports ADC and DAC modules up to 5 GHz, a very fast virtual FPGA pin structure and throughput at speeds of 20 Gb/s. It supports the Nector OS, which is a distributed C-based operating system. This OS allows communication between the FPGA and interface, direct real-time debugging capabilities, and direct high-speed bidirectional communication. It has built-in support for several bus technologies, such as PCI Express, 10 GB Ethernet and UART. It provides layered access to user defined environments and can be integrated with Matlab, Simulink and Xilinx System Generator. Beecube has many products

based on this technology, but the miniBee is most suited for research applications. miniBee features a Xilinx Virtex-6 FPGA with LX550T for general applications and SX475T for wireless

applications. It can support upto 6,400 Mb/s throughput per channel. It features an Intel Quad-Core i7 CPU. From a bus technology perspective, it supports the PCI Express Gen2 x4 interface to FPGA and PCI Express Gen2 x16 interface to the external connectors. Additional information on other features of the miniBee can be found in [Minibee, 2014]. BEE system technology has been cited by over 50 research papers, in a wide range of applications including wireless communications, HD video processing, signal intelligence, medical imaging and more.

Ettus Research [Ettus, 2014] provides software defined radio platforms such as the Universal Software Radio Peripheral (USRP) family of products. It supports a frequency range from DC to 6 GHz, including multiple antenna (MIMO) systems. Some application areas include white spaces, mobile phones, public safety, spectrum monitoring, radio networking, cognitive radio, satellite navigation, and amateur radio. The USRP X-series and the USRP Embedded series products can be effectively used for cross layer protocol design. USRP X3xx features two RF daughter board slots that can support bandwidth upto 120 MHz. It also supports multiple high-speed interfaces such as Dual 10 Gigabit Ethernet and PCI Express, each supporting throughputs upto 200 MS/s in full duplex mode. The PCI Express bus interface also offers a low latency of 10 microseconds. The product has 1G DDR3 memory which provides additional buffering and data storage memory, through its flexible access through the FPGA reference design. USRP x300 also provides multiple synchronization options, such as GPS synchronized timing alignment, which is ideal for MIMO applications. Hardware extensibility is possible through an external GPIO connector. Researchers can extend the capabilities of the platform through an external JTAG adapter that allows easy download and debugging of new FPGA images. The USRP is widely used by scientific research community. A Google Scholar search for USRP citations in the last three year time frame, yielded 2,000 results.

4.5 Comparison of Existing Systems using Proposed Framework

In this section, the relative merits and demerits of the six academic and commercial systems are analyzed using the evaluation framework described earlier. The first evaluation metric discussed is cost. As discussed in Section 4.2, cost refers to the overall cost of ownership. Since the general requirement is that the systems should be affordable, the value assigned to this metric is inversely proportional to the cost of the system. In other words, an expensive system will be assigned a lower value, whereas a low cost system will be assigned a higher value. Most of the commercial systems are feature rich, but they are generally designed out of reach of university researchers. Based on the pricing available on the Internet and through quotations received, a value of 7 has been assigned for the USRP x300, a value of 4 for the picoSDR, and a value of 3 for the miniBee. One of the advantages of these systems in the commercial category is accessibility, as they can be easily procured from the manufacturer. This may not be necessarily true in case of systems categorized in the academic category. Airblue is not available as a standalone product, so the only way one could perform research activity on Airblue is by building the board from scratch. While the design is readily available, there is significant amount of one-time cost involved in manufacturing the board. Unavailability of a printed circuit board schematic makes it difficult to achieve this easily. Due to this, a value of 4 has been assigned on the cost metric for Airblue. Hydra can be easily constructed using commercially available USRP boards, so a value of 7 has been assigned to it, same as the USRP. WARP boards are now available commercially from Mango Communications. But since these boards have been categorized under academic category, they have been evaluated considering the cost of building the hardware from scratch. There is plenty of information available from the research community on how to build the WARP boards. In addition, the components required for building this board are all commercially available at commodity prices. Due to this strong ecosystem, it is possible to build a board using the WARP design with minimal effort. Hence, a value of 6, between the value assigned for Airblue and Hydra, has been assigned to it.

Latency is the next evaluation metric that is discussed. Latency is the minimum time it takes to do a single transaction. This metric refers to the ability of the system to enable researchers to prototype algorithms with deterministic time constraints. One of the key enabling ingredients is the presence of a real-time operating system based processor. None of the systems described in this thesis have this feature. Having said this, Airblue, WARP, picoSDR, miniBee, and USRP x300 all feature a FPGA which enable this capability with careful programming. Hence, a value of 5 has been assigned to this metric. Since Hydra purely relies on a general purpose processor and cannot guarantee deterministic processing times, a value of 3 has been assigned to it. Newer application areas such as cyber-physical systems, which lead to convergence of control, communications and computing technologies, will drive the need for low latency systems [Kim and Kumar, 2012].

All the systems described in this thesis either support a PCI Express, a USB, or Gigabit Ethernet bus from the hardware to the computer. As shown in Figure 4.9, PCI Express has faster data transfer as compared to Gigabit Ethernet. Since PCI Express bus is the state of art for bus technology at time of publication, a value of 10 has been assigned to the Throughput metric to those systems that provide PCI Express as bus technology. This includes all the systems in the commercial category such as the picoSDR, miniBee, and the USRP x300. Hydra supports a USB 2.0 interface to the PC. USB 2.0 supports data transfer rates of 10s MB/s, which is order of magnitude slower than PCI Express. Due to this, a value of 5 has been assigned to this metric. WARP and Airblue both support a 10/100 Ethernet port, which has a data transfer rate between that of PCI Express and USB 2.0. Hence, a value of 7 has been assigned to this vector for these systems. It may be instructive to note that PCI Express bus technology, with its x4 and x16 variants, provides high throughput with low latency values. Hence, a combination of a real-time processor with PCI Express bus may enable designers to build a system that scores well on both the latency and throughput vectors.

Next, the metric of hardware agility is discussed. This refers to the ability of the system to reconfigure parts of RF, analog, and digital front-end during runtime. Runtime reconfigurability implies that parameters such as frequency and power of the RF front end can be changed, while running the rest of the protocol. Prototyping of cognitive radio algorithms can benefit from such a feature. To allow this feature, the system should have connectivity between these components and the FPGA fabric. Airblue and USRP x300 offer access to the RF front end through the FPGA logic, which allows for frequency reconfiguration. But they allow this only over two frequency bands, supporting the ISM applications. Additionally, they do not provide access to analog and digital pins on the board. Hence, a value of 6 has been assigned to this system, on the hardware agility metric. Both Hydra and WARP do not allow any type of reconfigurability of the analog and digital inputs. Most of the hardware reconfiguration on these two boards is through control from the general purpose processor. Due to this, a value of 3 has been assigned to these systems for this vector. While both the miniBee and picoSDR do not offer direct control of the RF front end through the FPGA logic, they provide an indirect mechanism to achieve the same. Additionally, they also have a continuously tunable radio front end with a frequency range up to 3 GHz. Hence, a value of 4 has been assigned to these systems for this metric.

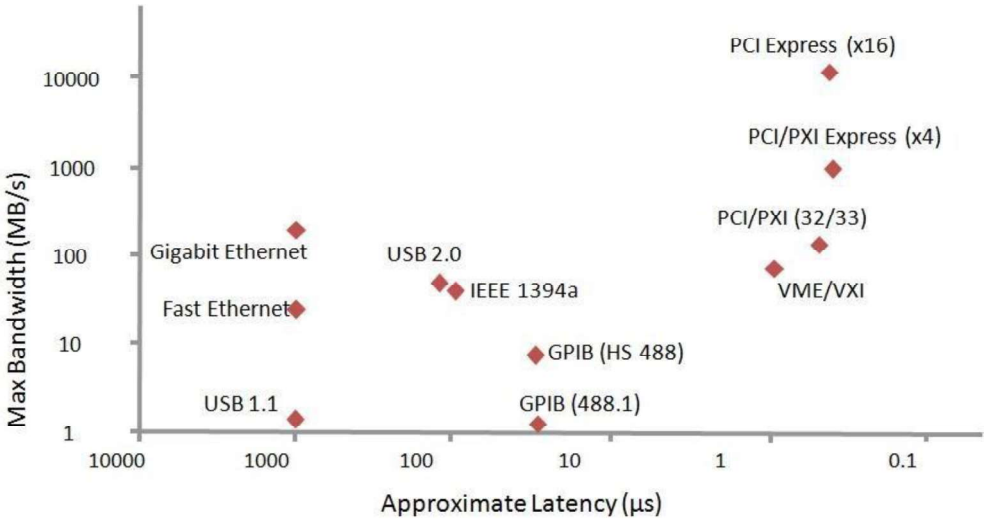

Figure 4.9: Comparison of latencies for different bus technologies

Software portability metric refers to the system's ability to allow users to extend research activities by easily porting software in and out of the system. In order to support this, it should be able to support different models of computation, such as state machine, dataflow, and text-based programming. This thesis shows that none of the systems described in this chapter offer support for all models of computation. Only the picoSDR supports both model-based design approach and text-based approach; hence it has been assigned a value of 6 for this metric. Other systems only support the text-based programming model. This allows text-based code to be ported to the system, as demonstrated by the popularity of GNU Radio community. Libraries available on this community can be easily ported to USRP and Hydra. Likewise, WARP offers enhanced support for porting text-based code across systems. Although these systems only support one model of computation, they have been able to foster community-based collaboration. Hence, they have been assigned a value of 4 for this metric. Since all the other boards support proprietary software and a single model of computation, a value of 2 has been assigned for the same.

The final metric relates to extensibility. This refers to the ability of the system to allow users upgrade the hardware and software functionality through well-defined interfaces. All the systems offer good interoperability with external modules such as Click Router module and NS2 through a standard software API. So, they generally score evenly on this front. However, their performance varies when it comes to hardware extensibility. The USRP x300 offers an external GPIO connector which researchers can use to extend hardware functionality through an external JTAG adapter. This provides superior flexibility; hence, a value of 8 has been assigned to this metric for the x300. miniBee, with its Honeycomb architecture, offers scalability to 4 FPGAs. Additionally, it offers flexible expansion options through the SFP/SFP+ and HDMI interconnects. Likewise, multi-channel ArgosV2 boards and Duarte's work on capacity analysis of full duplex systems have demonstrated WARP's capability for extensibility. Hence, a value of 6 has been assigned to both these systems for this metric. For all the other systems, a value of 3 has been assigned.

The resulting evaluation metrics spider-chart is shown in Figure 4.10. This evaluation is as per the specifications available at the time of publication of this thesis. As time progresses, the examples used in this thesis may evolve. While no system is expected to score a perfect 10 on all

vectors, such a comparison will offer an easy way of deciding trade-off points amongst these vectors. For example, cost and latency could be two very critical vectors enabling researchers in

developing nations to actively contribute to next generation communication system research. It is an open research topic to design an experimentation system that scores high on these metrics, while consciously trading off on throughput and extensibility metrics.

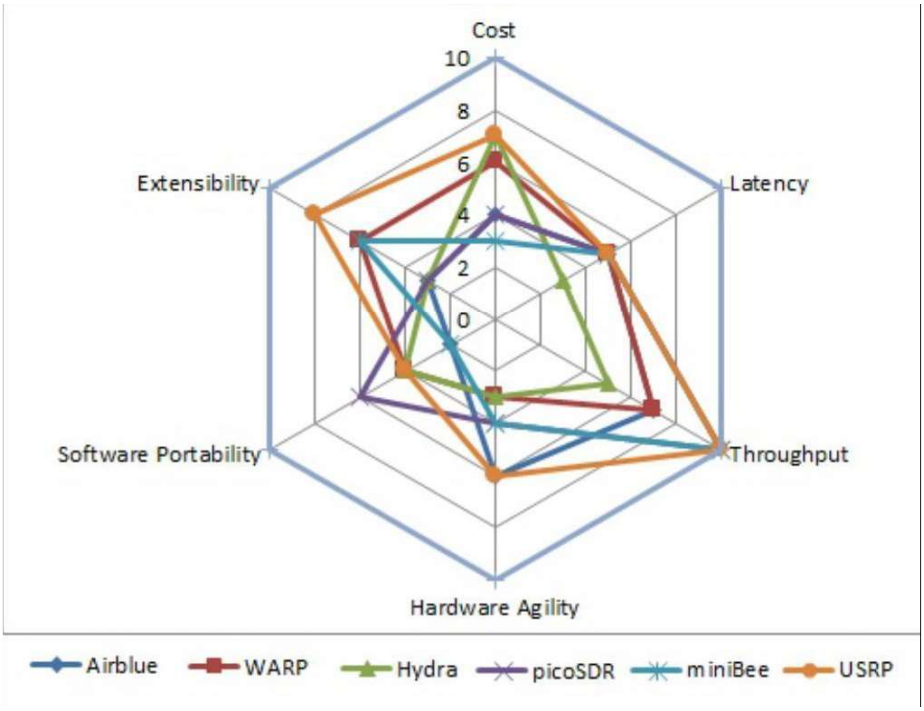

Figure 4.10 : Evaluation Metrics in a Spider Chart format

Table 4.1: Evaluation Metrics in a Tabular format

	Airblue	*WARP*	*Hydra*	*picoSDR*	*miniBee*	*USRP*
Cost	4	6	7	4	3	7
Latency	5	5	3	5	5	5
Throughput	7	7	5	10	10	10
Hardware Agility	6	3	3	4	4	6
Software Portability	2	4	4	6	2	4
Extensibility	3	6	3	3	6	8

4.6 Summary

Active research is being conducted on numerous new cross layer design protocols. The

underlying idea is to modify the traditional 7-layer OSI model and share more information across layers to achieve better throughput, bandwidth and energy optimization. As new information is available to different layers, there might be a need to reconfigure or make some changes to these

layers. However, such cross layer changes have to be done with caution as they can cause unintended consequences. Due to this, every change that is made has to be evaluated carefully to understand the overall system impact. Many cross layer design prototyping and validation systems are being developed in literature. Three systems in the academic category and three systems in the commercial category have been described in this chapter. The hardware architecture and key research findings that have been enabled by these systems have been outlined. An evaluation framework with six key metrics is defined, to evaluate these systems. This framework should serve as the basis for evaluating new experimentation systems that allows engineers and scientists to expand the scope of cross layer design research, without affecting any legacy systems. It furthermore offers a scientific approach for making trade-off decisions among the different vectors.

…